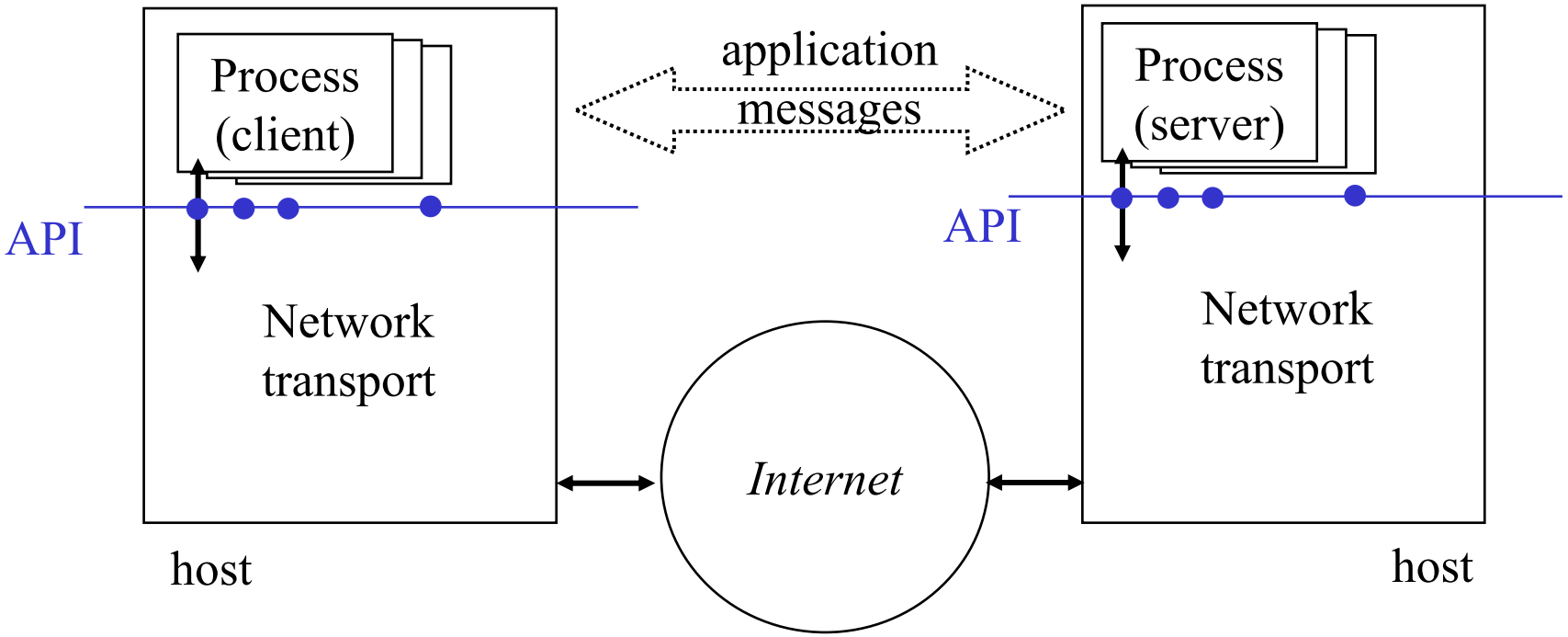
Network Applications

- **Consist of processes running on different host machines**
- **Require certain quality of services from network transport**
  - data loss, bandwidth, and timing



Internet Transport Services

- **Reliable service**
  - implemented by Transmission Control Protocol (TCP)
  - no missing or extra bits; in-order delivery
  - connection-oriented
  - no guarantee on bandwidth or timing
- **Datagram service**
  - implemented by User Datagram Protocol (UDP)
  - no guarantee on any quality of service
  - minimum overhead

Sample user network application

- **Fortune Cookie Service**
  - implemented in Java
  - complete source code available on the course Web site

Self study: download the code, and modify the code so that (1) the server and each client may run on separate host, and (2) the server listens to port 2888 instead.

Application-Layer Internet Protocols

- **File Transfer Protocol (FTP) ---** Self-study: try it if you have not done so
- **Telnet ---** Self-study: try it if you have not done so
- **HyperText Transfer Protocol (HTTP)**
  - used between a Web server and its clients
  - request (GET) / response (OK) messages
- **Simple Mail Transfer Protocol (SMTP)**
  - used on sending side: sender → sender's mail server → recipient's mail server
  - mail access protocols: recipient ←→ recipient's mail server

    Post Office Protocol (POP)

    Internet Mail Access Protocol (IMAP) – able to keep mailbox on the server
- **Simple Network Management Protocol (SNMP)**
- **Domain Name Service (DNS)**

HTTP Protocol

- **ASCII-based messages**

```
START_LINE <CR><LF>
HEADER_LINE <CR><LF>
……
HEADER_LINE <CR<LF>
<CR><LF>
ENTITY_BODY <CR><LF>
```

- **Persistent connections**
- **Cookies**
- **Web caches**

HTTP Request Messages

- **START_LINE to specify type of request**
  - most common: GET – fetch, HEAD – inquire status
  - others: OPTIONS, POST, PUT, DELETE, TRACE, CONNECT
  - objects referenced via their Uniform Resource Locators (URLs)
- **Example request**

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr
<CR><LF>
```

HTTP Response Messages

- **START_LINE to specify status of request**
  - common status codes: 200 – OK, 301 – Object moved permanently
    400 – Bad request, 404 – Object not found
- **Example response**

```
HTTP/1.1 200 OK
Connection: close
Date: Mon, 09 Jul 2000 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last Modified: Mon, 22 Jun 2000 09:23:24 GMT
Content-length: 6821
Content-type: text/html
<CR><LF>
( data data data… )
```

Persistent Connections

- **HTTP/1.0: uses a separate TCP connection for each data item**
  - inefficient because of TCP connection setup and teardown overhead
  - e.g., 13 connections have to be established and closed for a Web page with some text and a dozen of icons and small graphics
  - problem alleviated somewhat by parallel TCP connections
- **HTTP/1.1: allows one persistent connection for multiple request/response messages**
  - reduce the number of TCP connections that the server has to manage
  - requests may be pipelined to reduce response time
  - tradeoff: when to close the connection?

Use of Cookies

- **Identify user without prompting for username/password**
  - Associate user with an integer handle called cookie
- **Server assigns cookie to client in one of responses**
  - HEADER_LINE:

    Set-cookie: 1678455
- **Client includes cookie in future requests to server**
  - HEADER_LINE:

    Cookie: 1678455

Conditional GET

- **Support caching on local machine**

Client:

```
GET /fruit/kiwi.gif HTTP/1.0
Accept: text/html, image/gif, image/jpeg
```

Server:

```
HTTP/1.0 200 OK
Date: Wed, 12 Aug 1998 15:39:29
Last-modified: Mon, 22 Jun 1998 09:23:24
Content-type: image/gif
(data, data data, ....)
```

Client caches object & modification time

Client:

```
GET /fruit/kiwi.gif HTTP/1.0
Accept: text/html, image/gif, image/jpeg
If-modified-since: Mon, 22 Jun 1998 09:23:24
```

Server:

```
HTTP/1.0 304 Not Modified
Date: Wed, 19 Aug 1998 15:36:29
(empty body)
```

Client displays cached object

Web Cache

- **Also called Proxy Server**
  - sits between browser client and Web server
    - attempts to satisfy client's request with local copy
    - requests object from Web server when local copy is missing or has expired
- **Advantages**
  - better client response time
  - lighter Web server load
- **Cooperative Caching**
  - hierarchical organization of caches to distribute load
  - clustering of small caches at one site to handle large load

Interactive HTTP Session

- **Use telnet to open a TCP connection to a particular port**
  - Web servers use TCP port 80, so type following

    telnet <server host name, e.g., www.yahoo.com> 80
- **Type in HTTP Request message**
  - For example,

    GET /index.html HTTP/1.1

    host: www.yahoo.com

    <CR><LF>
- **Inspect Response message from server**
- **Repeat last two steps**